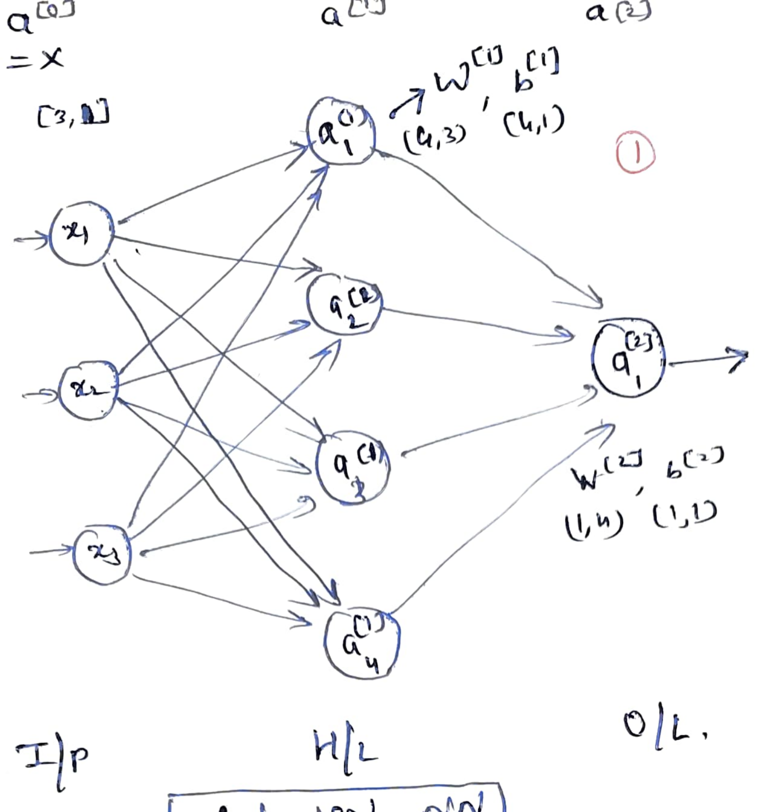
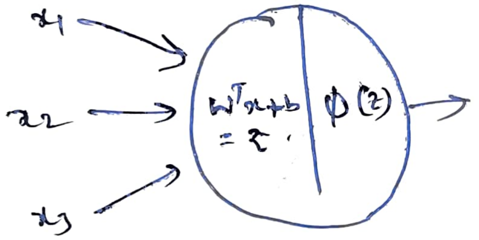
- ANN were first introduced in 1943 by the neurophysiologist Warren McCulloch & the Mathematician Walter Pitts.
- The model was based on Propositional Logic
- In 1960s it became clear that we would not be able to build truly intelligent machines.
- But in early 1980s, new architectures were invented and better training

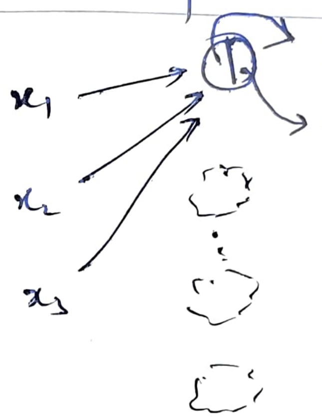
① 

②

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix}$$

2 Layer NN

③ 

④ 

$$z_1^{[1]} = W_1^{[1]T} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$a_i^{[l]}$ → $[l]$ → layer no.
i → node in layer

⑤

$$z_1^{[1]} = W_1^{[1]T} \cdot x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = W_2^{[1]T} \cdot x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = W_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = W_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

⑦

$$a^{[1]} = \begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} = \sigma(z^{(1)})$$

⑧

$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{x} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \underset{(4,1)}{\sigma(z^{(1)})}$$

$$\underset{(1,1)}{z^{[2]}} = W^{[2]}_{(1,4)} a^{[1]}_{(4,1)} + b^{[2]}_{(1,1)}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma(z^{[2]})_{(1,1)}$$

⑥

$$z^{[1]} = \underset{(4\times3)}{\begin{bmatrix} — W_1^{[1]T} — \\ — W_2^{[1]T} — \\ — W_3^{[1]T} — \\ — W_4^{[1]T} — \end{bmatrix}} \underset{(3\times1)}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} + \underset{(4,1)}{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}} = \begin{bmatrix} W_1^{[1]T} x + b_1^{[1]} \\ W_2^{[1]T} x + b_2 \\ W_3^{[1]T} x + b_3 \\ W_4^{[1]T} x + b_4 \end{bmatrix} = \begin{bmatrix} z_1^1 \\ z_2^1 \\ z_3^1 \\ z_4^1 \end{bmatrix}$$

Vectorizing for multiple Examples

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$

$a^{[2](i)}$ — $[2]$ ↑ layer 2; (i) ← example i

for $i = 1$ to m

$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$

$a^{[1](i)} = \sigma(z^{[1](i)})$

$z^{[2](i)} = W^{2} a^{[1](i)} + b^{[2]}$

$a^{[2](i)} = \sigma(z^{[2](i)})$

Using loop.

Vectorizing

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad (n_x, m)$$

$Z^{[1]} = W^{[1]} X + b^{[1]}$

$A^{[1]} = \sigma(Z^{[1]})$

$Z^{[2]} = W^{2} A^{[1]} + b^{[2]}$

$A^{[2]} = \sigma(Z^{[2]})$

$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

↔ Training Example

↕ Hidden unit No. In the layer

FWD & BWD steps Code

$dz^{[2]} = a^{[2]} - y$

$dW^{[2]} = dz^{[2]} a^{[1]T}$

$db^{[2]} = dz^{[2]}$

$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$

$dW^{[1]} = dz^{[1]} x^T$

$db^{[1]} = dz^{[1]}$

$dZ^{[2]} = A^{[2]} - Y$

$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$

$db^{[2]} = \frac{1}{m}$ np.sum($dZ^{[2]}$, axis=1, keepdims=True)

$dZ^{[1]} = W^{[2]T} dZ^{[2]} \cdot g^{[1]'}(Z^{[1]})$

$(n^{[1]}, m)$

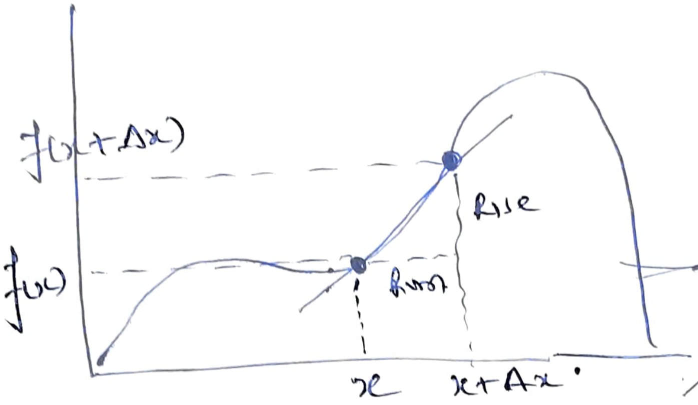
$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$

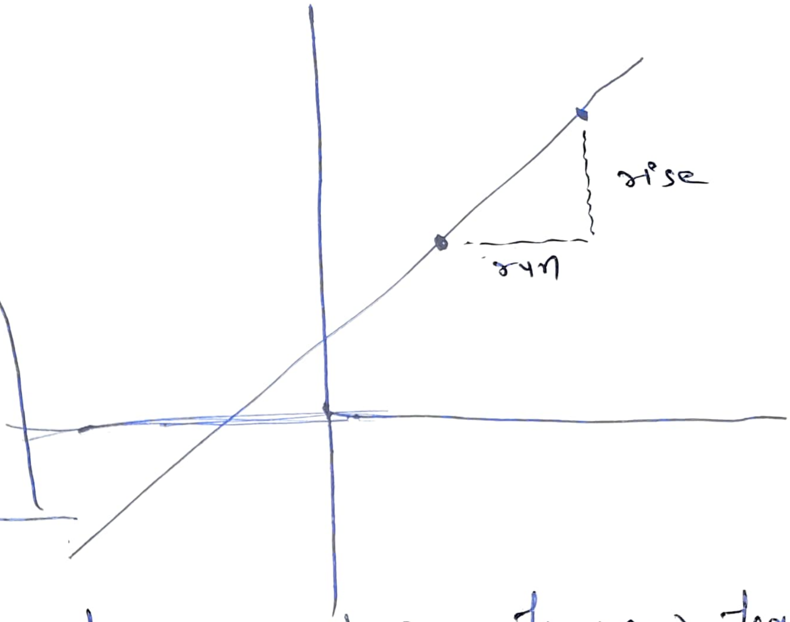
$db^{[1]} = \frac{1}{m}$ np.sum($dZ^{[1]}$, axis=1, keepdims=True)

Back propogation Intution

- For many years researchers struggled to find a way to train MLPs, without success.
- But in 1986, David Rumelhart, Geoffrey Hinton, and Ronald Williams published a ground breaking paper that introduced the "back propagation" training algorithm, which is still used today.
- It is a Gradient Descent using an efficient technique for computing the gradients automatically. in just two passes through the network (one forward, one backward), the BPA is able to compute the gradient of the network's error with regard to every single model parameter. In other words, it can find out how each connection weight and each bias term should be tweaked in order to reduce the error.
- Once it has these gradients, it just performs a regular GD step, and the whole process is repeated until the network converges to the solution.
- Automatic computing gradients is called "AUTODIFF".
- Reverse mode AUTODIFF is used by BPA.
- It is fast and precise, & well suitable when function to differentiate has many variables. (eg. connection weights) & few output (eg. one loss).

$$\text{Gradient} = \frac{\text{rise}}{\text{run}}.$$





$$\text{Gradient at } x = \frac{\text{Rise}}{\text{Run}} = \frac{f(x+\Delta x) - f(x)}{(x+\Delta x) - x} = \frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

→ Smaller Δx becomes (gives) better approximation

→ $$\frac{df}{dx} = f'(x) = \lim_{\Delta x \to 0} \left(\frac{f(x+\Delta x) - f(x)}{\Delta x} \right)$$

→ at small Δx. we call this differentiation.

$$f(x) = 3x + 2.$$

$$f'(x) = \lim_{\Delta x \to 0} \left(\frac{3(x+\Delta x) + 2 - (3x+2)}{\Delta x} \right)$$

$$= \lim_{\Delta x \to 0} \frac{3x + 3\Delta x + 2 - 3x - 2}{\Delta x} =$$

$$= \lim_{\Delta x \to 0} \frac{3\Delta x}{\Delta x}$$

$$f'(x) = 3$$

① Sum Rule.

$$\frac{d}{dx}(f(x)+g(x)) = \frac{df(x)}{dx} + \frac{dg(x)}{dx}.$$

② $f(x) = 5x^2$.   **Power Rule.**

$$f'(x) = \lim_{\Delta x \to 0}\left(\frac{5(x+\Delta x)^2 - 5x^2}{\Delta x}\right).$$

$$= \left(\frac{5x^2 + 10x\Delta x + 5\Delta x^2 - 5x^2}{\Delta x}\right).$$

$$= \lim_{\Delta x \to 0}\left(10x + 5\Delta x\right)$$ → $5\Delta x$ negligible.

$f'(x) = 10x$.

If $f(x) = ax^b$ then $f'(x) = abx^{b-1}$

③ $f(x) = \frac{1}{x}$

$$f'(x) = \frac{\frac{1}{x+\Delta x} - \frac{1}{x}}{\Delta x}$$

$$= \frac{\frac{x - x - \Delta x}{x(x+\Delta x)}}{\Delta x}$$

$$= \frac{\frac{-\Delta x}{x(x+\Delta x)}}{\Delta x}$$

$$= \frac{-1}{x^2 + x\Delta x}$$

$$= \frac{-1}{x^2}$$

Leonhard Euler.

$e^x = e^x$.

$f(x) = f'(x)$

$f(x) = \sin(x)$

$f'(x) = \cos(x)$

$f''(x) = -\sin(x)$

$f'''(x) = -\cos(x)$

$f''''(x) = \sin(x)$

(3) Product Rule

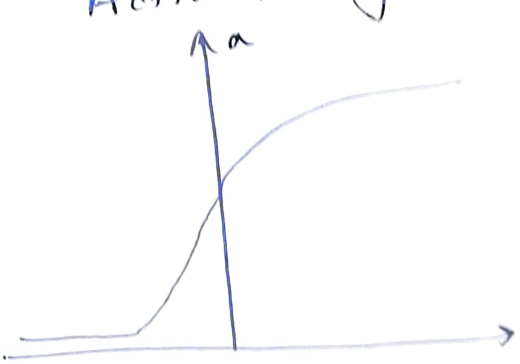
$A(x) = f(x) \cdot g(x)$

$A'(x) = f(x) \cdot g'(x) + g(x) \cdot f'(x)$

(4) If $h = h(p)$. and $p = p(m)$

then $\frac{dh}{dm} = \frac{dh}{dp} \times \frac{dp}{dm}$.

Activation function. — Sigmoid



$\frac{d}{dz} g(z)$ = slope of $g(x)$ at z

$= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$

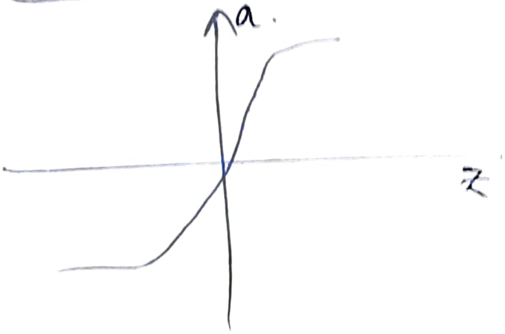
$= g(z)\,(1 - g(z))$

when z is large $\frac{d}{dz} g(z) = 0$

z is very small $\frac{\partial g(z)}{\partial z} = 0$.

$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$, when $g(z) = \frac{1}{2}$

$g'(z) = a(1-a)$ if $a = g(z) = \frac{1}{1-e^{z}}$



$g(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

$\frac{d}{dz} g(z) = g'(z)$ = slope of $g(z)$ at z

$= 1 - (\tanh(z))^2$

z large $\rightarrow \tanh(z) = 1$

$g'(z) = 0$.

z is small $\rightarrow \tanh(z) \approx -1$

$g'(z) = 0$.

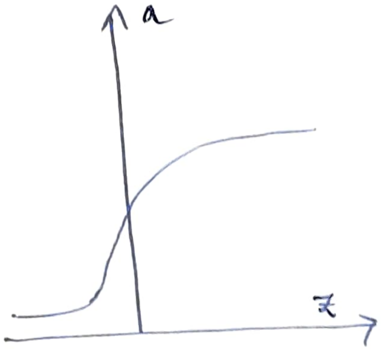
$z = 0$. $\tanh(z) = 0$

$g'(z) = 1$

Activation Functions

Sigmoid:

$$g(z) = \frac{1}{1+e^{-z}}$$



$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z$$

$$= \frac{d}{dz}(1+e^{-z})^{-1}$$

$$= -(1+e^{-z})^{-2} \frac{d}{dx}(1+e^{-z})$$

Using reciprocal Rule

$$-u'(z) / u(z)^2 = -u(z)^{-2} \cdot u'(z)$$

$$= -(1+e^{-z})^{-2} \left[\frac{d}{dx}(1) + \frac{d}{dz} e^{-z} \right]$$

$$= -(1+e^{-z})^{-2} \cdot e^{-z} \cdot (-1)$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1 \cdot e^{-z}}{(1+e^{-z}) \cdot (1+e^{-z})}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}+1-1}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= -(1+e^{-z})^{-2} \cdot \frac{d}{dz} e^{-z}$$

$$= (1+e^{-z})^{-2} \cdot e^{-z}$$

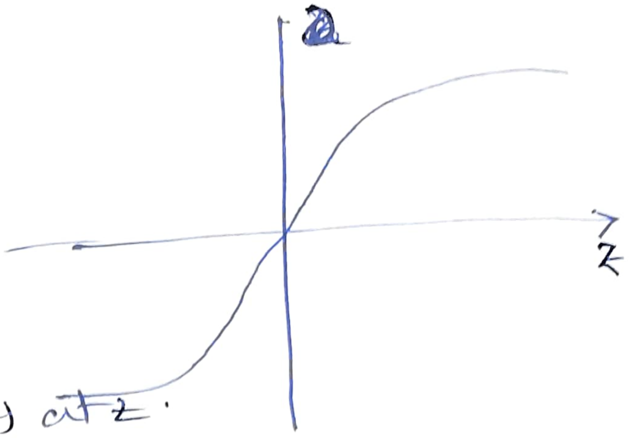
$$= \frac{1}{(1+e^{-z})} \cdot \frac{e^{-z}}{(1+e^{-z})}$$

$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}} \right)$$

$$= g(z)\,(1 - g(z))$$

$$g(z) = \tanh(z)$$

$$= \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z.$$

$$g(z) = \underbrace{(e^{z} - e^{-z})}_{f(z)} \cdot \underbrace{(e^{z} + e^{-z})^{-1}}_{y(z)}$$

$$g'(z) = f(z) \cdot y'(z) + y(z) \cdot f'(z)$$

$$f'(z) = \frac{d}{dz}(e^{z} - e^{-z}) = (e^{z} + e^{-z}) \rightarrow (1)$$

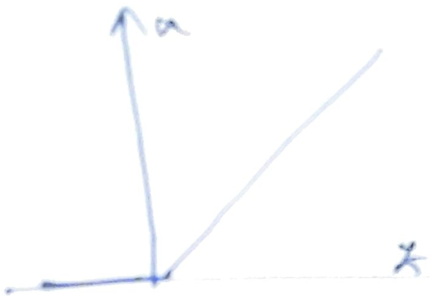
$$y'(z) = \frac{d}{dz}(e^{z} + e^{-z})^{-1} = -\frac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})^{2}} \rightarrow (2)$$

$$g'(z) = -(e^{z} - e^{-z}) \cdot \frac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})^{2}} + (e^{z} + e^{-z})^{-1} \cdot (e^{z} + e^{-z})$$

$$= \frac{(e^{z} + e^{-z})}{(e^{z} + e^{-z})} - \frac{(e^{z} - e^{-z})^{2}}{(e^{z} + e^{-z})^{2}}$$

$$\boxed{g'(z) = 1 - (\tanh(z))^{2}.} = \boxed{1 - a^{2}}$$

ReLU



$g(z) = max(0, z)$

$g'(z) = 0$ if $z<0$

$= 1$ if $z \geq 0$

$=$ und. if $z = 0$

Leaky ReLU.

$g(z) = max(0.01z, z)$

$g'(z) = 0.01$ if $z < 0$

1 if $z \geq 0$.

Back Propagation.

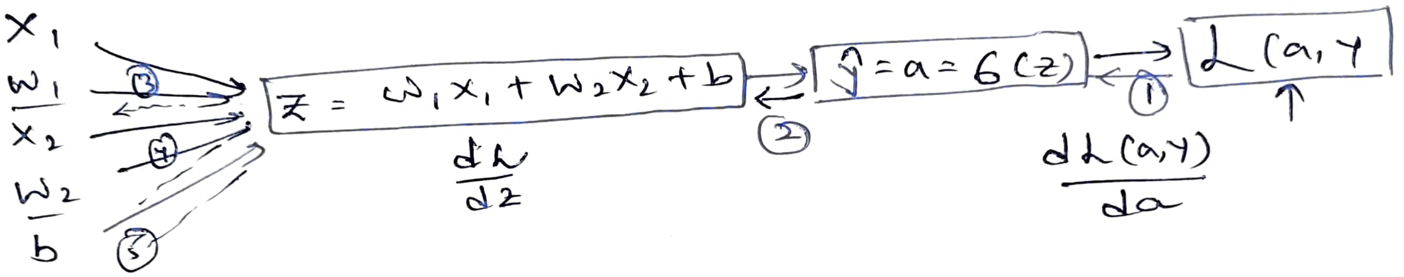
Logistic Regression Derivatives.

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -\left(y \log(a) + (1-y)\log(1-a)\right)$$

Forward step.



① $\dfrac{d\mathcal{L}(a,y)}{da} = \dfrac{-y}{a} + \dfrac{(1-y)}{1-a}$

② $\dfrac{dL}{dz} = \dfrac{dL}{da} \cdot \dfrac{da}{dz}$

$$= \left(\frac{-y}{a} + \frac{(1-y)}{1-a}\right) \cdot a \cdot (1-a) = a - y.$$

③ $\dfrac{dL}{dw_1} = \underbrace{\left(\dfrac{dL}{da} \cdot \dfrac{da}{dz}\right)}_{\frac{dL}{dz}} \dfrac{dz}{dw_1} = (a-y) \cdot \dfrac{d(w_1 x_1 + w_2 x_2 + b)}{w_1}$

$$= (a-y) \cdot x_1$$

④ $\dfrac{dL}{dw_2} = (a-y)\, x_2$

⑤ $\dfrac{dL}{db} = (a-y)$

Logistic Regression on m Examples

$x, W^{[1]}, b^{[1]} \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = \sigma(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \rightarrow \boxed{a^{[2]} = \sigma(z^{[2]})} \rightarrow \boxed{L(a^{[2]}, y)}$

($W^{[2]}, b^{[2]} \rightarrow z^{[2]}$)

$$\frac{dL}{dz^{[2]}} = a^{[2]} - y$$

$$\frac{dL}{dW^{[2]}} = \frac{dL}{dz^{[2]}} \cdot a^{[1]T} \Rightarrow \frac{dL}{dz} \cdot X,$$

$$\frac{dL}{db^{[2]}} = dz^{[2]}$$

$$\frac{dL}{dz^{[1]}} = \frac{dL}{da^{[1]}} \cdot \frac{da^{[1]}}{dz^{[1]}}$$

$$= \frac{dL}{da^{[1]}} \, g^{[1]'}(z^{(1)})$$

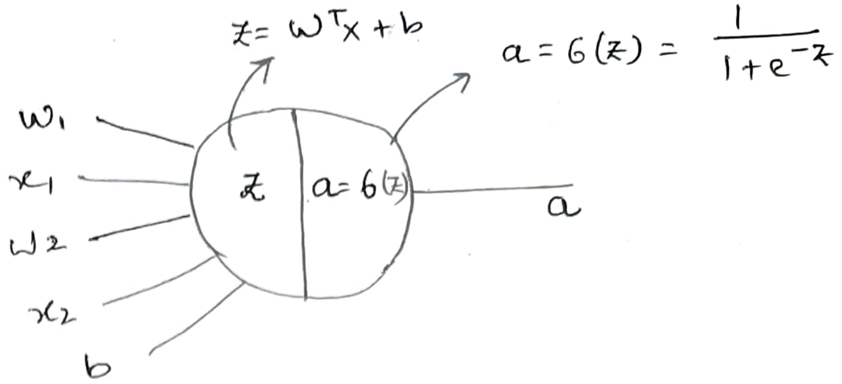
$$\frac{dL}{da^{[1]}} = \frac{dL}{dz^{[2]}} \frac{dz^{[2]}}{da^{[1]}}$$

$$\Rightarrow \frac{dL}{dz^{[2]}} * \frac{d(W^{[2]} \cdot a^{[l-1]} + b^{[l]})}{da^{[l-1]}}$$

$$\Rightarrow \frac{dL}{dz^{[2]}} \cdot W^{[2]}$$

$$\Rightarrow W^{[2]T} \cdot \frac{dL}{dz^{[2]}}$$

So. $$\frac{dL}{dz^{[1]}} = W^{[2]T} \cdot \frac{dL}{dz^{[2]}} * g^{(1)'}(z^{[1]})$$

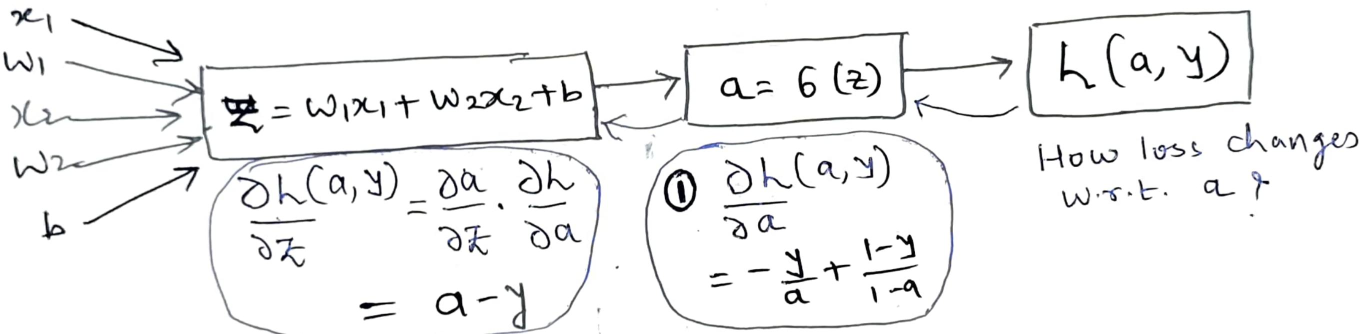


1) $z = w^T x + b$

2) $a = \sigma(z) = \hat{y}$

3) $L(y, a) = -[y \log a + (1-y) \log(1-a)]$

Computation Graph



① $\dfrac{\partial L(a,y)}{\partial a} = \dfrac{\partial -[y \log a + (1-y)\log(1-a)]}{\partial a}$

$= -\dfrac{y}{a} - \dfrac{(1-y)}{1-a} \dfrac{d}{da}(1-a) = \dfrac{-y}{a} + \dfrac{(1-y)}{1-a}$

② $\dfrac{\partial L(a,y)}{\partial z} = \dfrac{\partial a}{\partial z} \dfrac{\partial L}{\partial a} \Rightarrow \dfrac{\partial a}{\partial z} = \dfrac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$

$= a(1-a)$ → Proof on Backside

$\therefore \boxed{\dfrac{\partial L(a,y)}{\partial z}} = \dfrac{\partial a}{\partial z} \cdot \dfrac{\partial L}{\partial a} = a(1-a)\left[\dfrac{-y}{a} + \dfrac{(1-y)}{(1-a)}\right]$

$= -\dfrac{a(1-a) \cdot y}{a} + \dfrac{a(1-a)(1-y)}{(1-a)} = -(1-a) \cdot y + a(1-y) = -y + ya + a - ay$

$= \boxed{a - y}$

(3) $\frac{\partial L}{\partial w_1} = \frac{\partial z}{\partial w_1} \frac{\partial L}{\partial z} = \frac{\partial (w_1 x_1 + w_2 x_2 + b)}{\partial w_1} \cdot (a-y)$.

$$\boxed{\frac{\partial L}{\partial w_1} = x_1 \cdot (a-y)} \rightarrow (1)$$

Similarly

$$\boxed{\frac{\partial L}{\partial w_2} = x_2 \cdot (a-y)} \rightarrow (2) \quad \text{and} \quad \boxed{\frac{\partial L}{\partial b} = a-y} \rightarrow (3)$$

So, Eq^n (1), (2) & (3) are partial derivatives of L (loss) with respect to w_1, w_2, & b model parameters.

- It shows how L changes w.r.t. w_1, w_2, & b.

<u>Gradient Descent</u> (<u>for just one Example</u>)

Updates in next iteration.

$$w_1 = w_1 - \alpha \cdot \frac{\partial L}{\partial w_1} = w_1 - \alpha \cdot x_1 (a-y)$$

$$w_2 = w_2 - \alpha \cdot \frac{\partial L}{\partial w_2} = w_2 - \alpha\, x_2 (a-y)$$

$$b = b - \alpha \frac{\partial L}{\partial b} = b - \alpha (a-y)$$

Gradient Descent on m examples.

Cost function: $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(a^{(i)}, y^{(i)})$ → sum of individual losses.

$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$

$$\frac{\partial J(w,b)}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1} L(a^{(i)}, y^{(i)})}$$

For $(x^{(i)}, y^{(i)})$ pair

$\frac{\partial L}{\partial w_1}^{(i)}, \frac{\partial L}{\partial w_2}^{(i)}, \frac{\partial L}{\partial b}^{(i)}$

For one training example

Initialize $J = 0, \frac{\partial L}{\partial w_1} = 0, \frac{\partial L}{\partial w_2} = 0, \frac{\partial L}{\partial b} = 0$

For $i = 1$ to m

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J = J + (-1)[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})]$

$\frac{\partial L}{\partial z^{(i)}} = a^{(i)} - y^{(i)}$

$\frac{\partial L}{\partial w_1} += x_1^{(i)} \cdot \frac{\partial L}{\partial z^{(i)}}$ — For $n=2$

$\frac{\partial L}{\partial w_2} += x_2^{(i)} \frac{\partial L}{\partial z^{(i)}}$

$dw = dw + x^{(i)} dz^{(i)}$

$\frac{\partial L}{\partial b} += \frac{\partial L}{\partial z^{(i)}}$

$J = J/m, \quad \frac{dL}{\partial w_1} = \frac{\partial L}{\partial w_1}/m, \quad \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial w_2}/m, \quad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial b}/m$

Now:

$w_1 = w_1 - \alpha \cdot \frac{\partial L}{\partial w_1}$

$w_2 = w_2 - \alpha \cdot \frac{\partial L}{\partial w_2}$

$b = b - \alpha \cdot \frac{\partial L}{\partial b}$

This is one step of gradient descent Algorithm.